Extensible Middleware Framework for Multimodal Interfaces in Distributed Environments

Vitor Fernandes[1], Tiago Guerreiro[2], Bruno Araújo[3],Joaquim Jorge[4], João Pereira[5]
IST-UTL / INESC-ID
R. Alves Redol
1000-029 Lisbon, Portugal
+351 21 3100363
{vmnf[1], tjvg[2], brar[3]}@immi.inesc.pt, jaj@acm.org[4], jap@inesc.pt[5]

ABSTRACT
We present a framework to manage multimodal applications and interfaces in a reusable and extensible manner. We achieve this by focusing the architecture both on applications' needs and devices' capabilities. One particular domain we want to approach is collaborative environments where several modalities and applications make it necessary to provide for an extensible system combining diverse components across heterogeneous platforms on-the-fly. This paper describes the proposed framework and its main contributions in the context of an architectural application scenario. We demonstrate how to connect different non-conventional applications and input modalities around an immersive environment (tiled display wall).

Categories and Subject Descriptors
H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces – *Collaborative Computing, Computer-supported cooperative work, Synchronous interaction;*
H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *Input devices and strategies, Interaction styles;*
H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems – *Artificial, augmented, and virtual realities.*

General Terms
Algorithms, Design, Human Factors.

Keywords
Framework, Multimodal Interfaces, Extensible, Reusable, Capabilitie, Collaborative.

1. INTRODUCTION
Traditional interface devices in the HCI (Human Computer Interaction) area, basically the mouse and the keyboard, are still overwhelming before the emergent ways of interaction. However, in the last decades great efforts were made with promising results to present interaction options between persons and any type of computer. These alternative devices can replace, extend or completely change the interaction between that person or group of persons and a computer or group of computers.

At our immersive laboratory, LEMe Wall [1], we have several researchers studying different areas, some of them complementary

1.1 LEMe Wall
LEMe Wall is an intelligent distributed environment with a multi-projection tiled display wall as the main component. It is composed by three essential modules: a 4x3 projectors matrix duly supported a flexible screen that offers visualization support and a computer cluster that controls the projection. The environment is complemented by a set of sensors and actuators that increase the interaction immersion and naturalness.

1.2 Motivation
At the Intelligent Multimodal Interfaces Group, INESC-ID, we have several junior and senior researchers leaning over several prototypes across different working areas (Modeling, Visualization, Interaction, Mobility, Accessibility …). Most of the times, researchers learning over a certain area have to replicate others' work on different areas and create a whole new demonstrable prototype. In an environment where several multimodal applications are used, the developers' effort is often wasted due to its rigid focus on a certain application. The cycle continues and every new prototype is a new whole not-reusable product. Within a research group, we can easily find prototypes and solutions that became unusable and therefore useless to other researchers.

Observing these scenarios it was urgent to find a suitable solution that: i) make it possible for any researcher to focus on a determined module, using already developed modules to complement and demonstrate his accomplishments; ii) can be used across heterogeneous platforms making it versatile considering developers profile and needs; iii) Provides the user on-the-fly module management so all the resources can be maximized and used when needed the most.

2. RELATED WORK
Trying to overcome the stated integration difficulties, in a first approach, we surveyed existent platforms. We looked into several areas including multimodal interfaces, multi-agents communication frameworks, capabilities, matchmaking that

ICMI'07, November 12–15, 2007, Nagoya, Japan.

intersect our work on Extensible Middleware Framework.

Quickset [3] compared speech-pen interfaces vs. pen only and speech only on a collaborative environment using handheld and desktop PCs. Both modalities strengths and handicaps are stated: speech is bad for line drawing and excellent for selecting objects not visible on screen, pen is the reverse. Thus multimodal interaction achieves fast task performance against unimodal interfaces.

Open Agent Architecture (OAA) [8] from SRI is a multiple agent environment. Application agents and user interface agents, use a facilitator to establish communication using an Inter-agent Communication Language (ICL). Facilitator maintains a knowledge base that records the capabilities of a collection of agents and uses that knowledge to assist requesters and providers of services in making contact. Multiple interaction modalities can be used and joined through user interface agent.

RETSINA Communicator [10] is a Multi-Agent System (MAS) providing abstraction that supports communication between agents. Client / server and peer to peer connections are available to agents. RETSINA supports **KQML** [4] Agent Communication Language (ACL).

Framework for Rapid Development of Multimodal Interfaces [6] tries to overcome the limitation of prior frameworks (there are few commercial multimodal applications and also require a lot of development time and resources) enabling rapid development of applications using multiple modalities with a novel multimodal fusion. A fusion algorithm obtains data from modalities, fuses that data to a meaning and takes action based on that meaning as done on Put-that-there [2].

A Real-Time Framework for Natural Multimodal Interaction with Large Screen Displays [9], where gestures and speech modalities receive adequate and timely feedback from a large screen display that as high demand on Human-Computer Interaction (HCI) that should be effortless and natural to users.

Although the reutilization and modality integration issue has been studied by the mentioned researchers, we still can't find a solution that easily provides the developer an easy platform to join and demonstrate his module, and provides the user with on-the-fly module integration.

3. EXTENSIBLE MIDDLEWARE FRAMEWORK

The main goal of our framework is to manage input modalities and applications separately allowing that each component can be reused and extended. We focus on inputs' capabilities and application' interests offering at each moment the most suited input for a determined task in a specific application. For a given input, capabilities are the set of tokens or input data that can be offered by the input to any application, i.e., mouse commands, gesture tokens, tracking positioning or speech commands. On the other hand, a given token can be delivered by several input types, for example an "up direction" command could be given by a keyboard input or tracked body gesture motion. Our framework provides the capacity to manage input modalities and capabilities accordingly with the application's will. We focus on reusing input capabilities and being able to add new modalities and applications with a few amount of effort. On the other hand, focusing our architecture both on devices' Capabilities and users' Intentions, we enable its use in collaborative environments, where any input is managed accordingly with the application and the capabilities available at the given moment. The framework is responsible to redirect needed tokens from the inputs to the application according to its preferences. Following this, different modalities can be integrated and shared between applications. Moreover, a new input which is able to deliver a known token for an application can be integrated easily extending the modalities without any change on the application.

3.1 System Overview

In order to accomplish our extensibility goals, we used a message-oriented approach. Our system architecture, such as depicted by Fig. 1, is organized into four different entities: **Input**s, **Application**s, a **Manager** and a communication backbone called

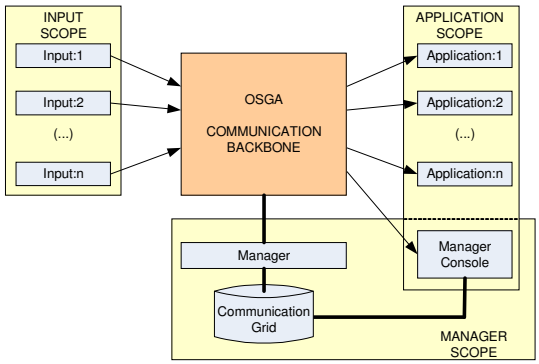

Figure 1 - System Architecture

OSGA.

The Inputs are the communication interfaces for devices such as keyboard, mouse, speech recognition system, EMG (electromyographic signal capture), camera tracking systems which are able to deliver multimodal tokens corresponding to gestures commands or even data files. Applications are linked to the communication backbone through an interface to allow the interaction using Inputs information. The communication backbone is responsible to redirect all the messages, allowing multiple Inputs to interact with multiple Applications. Doing so, Inputs can publish their data that will be received by all applications which have subscribed to it. The communication and the inputs' discovery process are coordinated by a Manager. This module supports the configuration and establishment of communications between Inputs and Applications. The coordination relies on a connection grid mechanism which stores the capabilities available for each connected input and the needs or preferences of each application. Finally, Manager offers a user interface which allows the user switching and choosing inputs for applications. This interface takes advantage of existing inputs such as Applications do.

3.2 Open Source Groupware Architecture

Having an overwhelming number of prototypes emerging from investigation work, enables one to devise future integration projects bringing together several pieces to create larger and integrated applications. **Open Source Groupware Architecture (OSGA)** [8] is a distributed XML message-based integration framework developed within our research group to overcome the

integration problem. This framework can be easily used for further applications and can be integrated in our current prototypes. It is built upon XmlBlaster [11] and provides the capacity to have several clients receiving messages accordingly to both subscription and publishing mechanisms. Basically, all the messages sent to the system are redirected to all the clients which subscribed to a given message topic. It also enables to filter the messages for a topic taking in account additional message properties such as sender's identification, or clues about the message content.

3.3 Communication Protocol

To support the interaction between Inputs, Applications and the Manager, we organized our XML messaging protocol (Fig. 2.) into four separated stages: i) initialization of Input; ii) initialization of Application; iii) communication setup and iv) communication suspension, resume and termination which are coordinated by Manager.

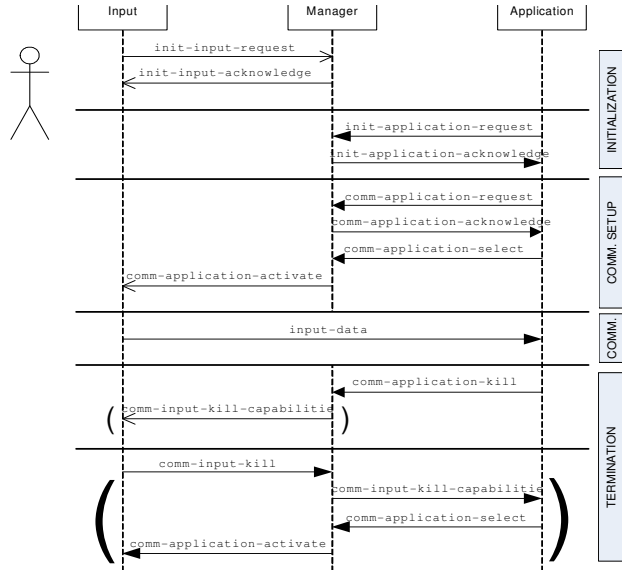

Figure 2. - Protocol Sequence Diagram

Initialization: The Input initialization is performed by an `init-input-request` message using a temporary communication channel. As illustrated in the following XML example, each input provides its identification and the list of supported capabilities. The identification is formed by its type (i.e. Speech, EMG, Tracker…) and the key that will be used by the manager to assign a global unique identifier (i.e. Speech:1) to Input. This information is stored on Manager, and the new identification is used by Input to create its own input channel which will be available for interested applications.

```
Input->Manager:Input
<input>
  <msg>init-input-request</msg>
  <name>Speech</name>
  <description>Speech Recognizer</description>
  <key>54JH6G34J5G4HJ76</key>
  <capabilities>
    <capability>
      <name>Token</name>
      <value>Hello World!</value>
    </capability>
    ...
```

Applications also require an initialization in order to be identified by the Manager. A list of needed capabilities is provided and stored in the *communication grid*. Manager defines a global identifier (example Paint:1) and provides a list of the most suited inputs to satisfy the capability needs by the application. The following XML message `init-application-acknowledge` presents a response. This information will be used by the application to establish the connection with needed inputs to support the multimodal interaction.

```
Manager -> Application:Temp (Filtered)
<manager>
  <msg>init-application-acknowledge</msg>
  <id>1</id>
  <input>
    <name>Speech</name>
    <description>Speech recognizer</description>
    <id>2</id>
    <capabilities>
      <capability>
        <name>Token</name>
        <value>Hello World!</value>
      </capability>
      ...
```

Communication and Interaction: Applications can request at any given time, the refereed list of inputs and capabilities, and the Manager will retrieve the actualized version of it. With the gathered information applications can select the desired capabilities accordingly to the users' preferences.

At this time, the user through application can select and pair the desired capabilities with the application possible actions. This information is kept in the application *translation matrix*. On the other hand, the list of needed capabilities is stored in the manager communication grid. Although the inputs' capabilities have some semantic meaning, the user is able to pair the capability with a completely different action (i.e. the Application receives the message UP which is translated to DOWN). This phase can be compared with keyboard calibration in a computer game. Manager receives this information and completes the communication grid with the subscribed inputs and capabilities.

It also replies to Inputs (the ones with requested capabilities) informing that Applications are interested on receiving their data. An Input only publishes data that has been requested! Each Application only receives what it revealed interest on through the use of filters (implemented in OSGA). The *communication link* is established between the Inputs and the Application that subscribed these Inputs' topics. With all the required topics subscribed we achieve a n:n communication between Inputs and Applications improved by restriction mechanisms (filters) and enriched descriptions (messages traded) that can be updated on the fly.

Direct-Connection: Data can be exchanged directly between Inputs and Applications (1:n) after socket negotiation (tcp or udp) without using Manager or OSGA allowing higher performance for multimedia applications or other.

Suspend and Resume: Communication between an Input and an Application can be suspended by Application initiative through the cancellation of Input subscription. To resume communication it will be enough to subscribe the topic again. In case the Input of which was suspended the communication finishes, the manager informs the Application as described previously.

Terminate: Applications can terminate their session at any time publishing that intention to the Manager that publishes this to

affected Inputs that must, eventually, reduce published data.
When an Input finishes the communication Manager publishes to affected Applications that they have to modify the received capabilities and also sends the current list of Inputs and respective capabilities for Application selection**.**
Keep-alive: To prevent failure situations when Inputs or Applications aren't able to communicate with Manager we developed a keep-alive signaling between this entities and the Manager. If a failure has been detected the termination protocol is followed depending on the "dead" entity.

4. APPLICATION CONTEXT

The use of the framework in our immersive environment makes possible for all input prototypes developed to be available to everyone. This is quite useful when we consider project integration and reusing others researchers effort. Actually, when any student / researcher design his project, he can visualize all the available inputs and capabilities through the Manager Console.

Collaborative Design Scenario. We present a scenario with two inputs and one application. Let's consider a collaborative design scenario where two users are preparing a mould at LEMe Wall. The selected inputs are the Tablet PC and a Laser pointer, operated by two users. The application used is Gides++ [6], an application developed within our research group, and the users are designing a mould. With our approach the users can operate within the same application improving collaboration and the design performance (Fig. 3). In this scenario, the user subscribed both input's capabilities receiving messages from both Inputs. Therefore, two users can work concurrently on one Application.

Figure 3. – Collaborative Design Scenario

Both users interact over the same application but they can be focused on different aspects, tasks or views.

5. FUTURE WORK

The next step in our work is to evaluate the framework considering two roles: the developer's role and the ease to add new Inputs and Applications; and the user's role considering control tasks and the performance obtained while executing them. Considering Inputs we will develop composed actions and commands focusing on multimodal parsing. To achieve this goal we will enrich our framework with Composed Inputs that subscribe several device messages, parse and disambiguate them and create higher level messages available to Applications.

6. CONCLUSIONS

We presented a framework to manage multimodal interfaces in a distributed environment focusing on the extensibility and reusability of input modalities. We achieve this goal by separating inputs and applications and managing their communication through a protocol over a message-based system. The main contributions of the presented idea are the possibility to adapt inputs and applications on-the-fly accordingly to the available capabilities and user's desire at any given time and allowing direct-connections between Inputs and Applications without Manager / OSGA after negotiation. Collaborative scenarios are well suited with this framework focus and goals. We presented an application scenario in an immersive environment where the collaborative and multimodal advantages are huge.

7. ACKNOWLEDGMENTS

This work was supported in part by European Commission grants IST-2003-004785 (IMPROVE) and IST-2006-034525 (SATIN).

8. REFERENCES

[1] Araújo, B., Guerreiro, T., Jorge, J., Pereira, J., Jota, R. Leme Wall: Desenvolvendo um sistema de Multi-projecção. Proceeding of 13º EPCG , Vila Real, Portugal, Oct 2005.

[2] Bolt, R. A., July 1980., Put-that-there: Voice and gesture at the graphics interface, in *Computer Graphics (SIGGRAPH'80 Proceedings)*, **14**(3):262–270.

[3] Cohen, P.R., Chen, L., Clow, J., Johnston, M., McGee, D., Pittman, J., and Smith, I., 1996, Quickset: A multimodal interface for the distributed interactive simulation, in *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'96)*, Demonstration Session, Seattle, WA.

[4] Fritzson, Rich, Finin, Tim, McKay, Don and McEntire, Robin, 1994, KQML - A Language and Protocol for Knowledge and Information Exchange, in *13th International Distributed Artificial Intelligence Workshop*, Seattle WA.

[5] Flippo, F., Krebs, A., and Marsic, I. 2003. A framework for rapid development of multimodal interfaces. In *Proc. of the 5t h ICMI* 03. ACM Press, New York, NY, 109-116

[6] Jorge, J.,Silva, N., Cardoso, T. Gides++, Proceedings of 12º EPCG, Porto, Portugal, Oct 2003.

[7] Martin, D. L., Cheyer, A. J., Moran, and D. B., 1999, The open agent architecture: a framework for building distributed software systems, in *Applied Artificial Intelligence*, vol. 13, nos. 1-2, pp. 21-128.

[8] OSGA, 2007, Open Source Groupware Architecture (April 5, 2007), http://www.osga.net.

[9] Krahnstoever, N., Kettebekov, S., Yeasin, M. and Sharma, R., 2002, A Real-Time Framework for Natural Multimodal Interaction with Large Screen Displays, in *Fourth IEEE ICMI 2002*.

[10] Shehory, O. and Sycara, K., 2000, The RETSINA communicator, in *Proceedings of the Fourth international Conference on Autonomous Agents* (Barcelona, Spain, June 03 - 07, 2000), AGENTS '00. ACM Press, New York, NY, 199-200.

[11] XMLBlaster, 2007, XMLBlaster (April 5, 2007); http://www.xmlblaster.org.