DI 1 - Common Concepts and Software Tools

A Report of the Esprit Project 8579 MIAMI – WP 1 –

April, 1995

Written by
L. Schomaker, J. Nijtmans (NICI)
A. Camurri, F. Lavagetto, P. Morasso (DIST)
C. Benoît, T. Guiard-Marigny, B. Le Goff,
J. Robert-Ribes, A. Adjoudani (ICP)
I. Defée (RIIT)
S. Münch (UKA)
K. Hartung, J. Blauert (RUB)

Contents

Chapter 1

WP 1: Taxonomy of Multimodal Interactions and Common Software Tools

Below is an *Overview of all DELIVERABLES*

D1: Common Software Tools for Multimodal Experiments *(after WP 1)* $\Longleftarrow$

D2: Progress Report *(after WP 2)*

D3: Basic Software Architecture *(after WT 3.3)*

D4: Completed Software Architecture *(after WP 3)*

D5: Symbolical Demonstrator *(after WP 4)*

D6: Analogical Demonstrator *(after WP 4)*

D7: Evaluation Report *(after WP 4)*

In this report, we will mainly present *Common Software Tools for Multimodal Experiments*. Although not defined as a deliverable, we would like to mention briefly another product of the Work Package 1: our Taxonomy Report in section 1.1. Then, we will introduce the actual part of this deliverable, the documentation of the software tools in section 1.2.

1.1 Taxonomy Report

1.1.1 Goals

This Workpart is an initial stage of the project, aiming at developing a common platform for partners' cooperation. In the initial phase, the known body of results about visual, acoustical, and haptic systems will be collected by partners, distributed and assessed. Next, the taxonomy of multimodal interactions with respect to information flow will be developed. This will be done by considering different levels of processing and integration.

It is well known that each of the subsystems of HIP can operate separately and built representations which are much related. For example, a representation of spatial relations can be built by inputs from visual, acoustical, and tactile systems. The visual input is clearly the richest, but the most important aspect here is that

all the representations are overlapping. In fact, there might be only a single common representation of basic spatial properties, relying on inputs from all the subsystems. When two or more systems cooperate, the representation is enhanced. Highly specialized systems operate to built such a representation (e. g. stereoscopy in vision and sound), which can also be memorized, recalled, and easily manipulated. One special aspect is conversion: a representation built by one system can be used by another one. For example, if a visual representation of a room is built, a person can move even in darkness using memory and a tactile system.

On a different level of processing there are representations which serve in human communication and are thus "symbolic". They cover an enormous range, e. g. from simple sounds to complex language expressions. These representations are learned, and the represented concepts can be associated with any type of inputs and outputs from the different HIP subsystems. This again suggests that there is one common representation that is built incrementally and relies on continuous enhancement by using different modalities.

The taxonomy built in this Workpart will be used for developing a general cybernetic model concentrated on multimodal interaction, a common representation, and memory. The model will serve as a basis for the development of experimental procedures with human subjects. From a technical point of view, the taxonomy and the model will be used to assess problems arising in practical realizations of multimodal data acquisition and representation in computer systems.

On the next page the original Work Task description is displayed.

TA Work Task Synopsis

ESPRIT BASIC RESEARCH Project 8579 MIAMI	WP No. 1	WT No. **1.1**
Task title: **Taxonomy** Partner responsible: NICI Start date: 01/01/94 End date: 31/03/94 Task manager: L. Schomaker Planned resources: NICI: 2 - DIST: 2 - ICP: 2 (in man-months) RIIT: 2 - UKA: 2 - RUB: 2		Sheet 1 of 1 Issue date: 08/10/93

Objective:

This work task concerns the definition of multimodal channels and their suitability for use in multimedia interfaces. State of the art literature of research in the different modalities is identified. The taxonomy is a concise hierarchical description of modalities in human-computer interaction.

Input:

State of the Art, Experiences and Taxonomy from other projects

Output:

Common Glossary & Taxonomy

Approach:

- Define the structure of the taxonomy
- Define the glossary of terms
- Identify state of the art research for each of the components in the taxonomy
- Clarify the subset of modalities covered by **MIAMI**

Contributions:

ALL state of the art expertise

1.1.2 Realization

The Taxonomy Report consists of a 187-page document. It serves as a basis for future work in the project. The basic terms which will be used in MIAMI will be defined and an overview on man-machine-interfaces will be given. The term "taxonomy" is used in the following sense, adapted from [2]: "1: the study of the general principles of scientific classification: SYSTEMATICS; 2: CLASSIFICATION; specif: orderly classification of plants and animals according to their presumed natural relationships"; but instead of plants and animals, we attempt to classify input and output modalities.

First we focus our attention on the human perceptual process, its input channels (HIC) and the characteristics of computer output media (COM) (Chapter 1).

Then, we will approach the process of human control and manipulation, dwelling on the characteristics of the human output channels (HOC) and the computer input modalities (CIM) (Chapter 2).

In the next chapter (3) the issue of the interaction process will be dealt with, addressing aspects of input/output coupling and temporal and spatial synchronization.

Gradually moving from basic processes to more abstract levels, we will describe aspects of cognition and learning, its representational architecture in machine and man, as far as possible, and on the issue of interacting agents (Chapter 4).

Finally, in the last chapter, a number of scenarios, or rather "dreams" will be depicted, elucidating fundamental aspects put forward in this taxonomy.

Availability

This Taxonomy Report is also available on the World-Wide Web under

`http://www.nici.kun.nl/~miami/taxonomy/taxonomy.html`

The original document is produced in LaTeX, and has been converted to HTML using the latex2html convertor. The resulting HTML infostructure looks a little less appealing than the original document, but has the advantage of allowing for updates as the project continues.

1.2 Common Software Tools

1.2.1 Goals

For Work Task 1.2, the development of common software tools was undertaken, in order to allow an easy exchange of experimental modules and obtain a common architecture. Orginally envisaged as a minor undertaking in the Technical Annex, this task turned out to be a crucial one as regards the feasability of the bimodal experiments to be performed in Work Package 2. Therefore, much effort has been spent on this tasks by all partners. As a consequence, work in the area of experimentation is somewhat delayed. At the same time, however, the software produced at this stage is an anticipation on Work Package 3, were the current efforts will pay off.

1.2.2 Realization

A number of levels can be defined, on which substantial and mostly **completely new software** development has taken place within MIAMI:

1. Inter-process communication and parallellism (2)
2. Two-dimensional graphics and widget control (3)
3. Three-dimensional graphics and widget control (4)
4. Audio rendering software (5)
5. Specific Experimental applications (6)

On the next two pages, the original Deliverable Description and a synopsis of the relevant Work Task are given. The resulting developed software will be reported in the following chapters.

DELIVERABLE DESCRIPTION SHEET

ESPRIT BASIC RESEARCH Project 8579 MIAMI	Deliverable No. 1
Name of deliverable: **Common Software Tools for Multimodal Experiments** Partner responsible: NICI Date of delivery: 31/08/94 Status of deliverable: Public	Sheet 1 of 1 Issue date: 08/10/93

Technical description:

On the basis of the common environment (SUN/Xwindow system) a library is developed for rendering polyhedrons and dynamically controlling their parameters in real time. This basic platform allows for the execution of the bimodal experiments under highly comparable conditions for the different bimodal experiments. The library will be written in C/C++.

Future use:

Will be used in most work tasks of WP 2 and will provide a starting point for the architecture design in WP 3

Form of presentation:

Documentation, dissemination of source code among partners

TA Work Task Synopsis

ESPRIT BASIC RESEARCH Project 8579 MIAMI	WP No. 1	WT No. **1.2**
Task title: **Experimental design** Partner responsible: NICI Start date: 01/03/94 End date: 30/06/94 Task manager: L. Schomaker Planned resources: NICI: 3 - DIST: 2 - ICP: 3 (in man-months) RIIT: 3 - UKA: 4 - RUB: 3		Sheet 1 of 1 Issue date: 08/10/93

Objective:

Definition of a common, and basic experimental platform which enables the study of bimodal *control* and *perception*. In case of bimodal *control*, the paradigm consists of a virtual world of polyhedrons. Both analog and symbolic control modes are addressed. Characteristics of the polyhedrons (shape, kinematics, sound) are controlled through two modalities. In case of bimodal *perception*, the same paradigm is used to test psychophysical aspects of spatial source localization.

Input:

WT 1.1

Output:

Experimental software for bi-modal experiments

Approach:

- Develop a basic Xwindows graphics program for rendering moving 3-D polyhedrons
- Define object characteristics (shape, kinematics, and sound)
- Define control modes and perceptual variables
- Basic documentation

Contributions:

ALL software; ALL documentation

Chapter 2

Inter-process communication and parallellism

2.1 Introduction

For MIAMI, a special shell, dubbed **M**wish, has been developed. This application is based on the wish-shell written by John Ousterhout [3], but extended with some MIAMI-specific additions. The additions are in the form of separate libraries. These can be linked statically or dynamically to Mwish. Only a small patch to C code of the standard wish shell is needed.

These are the libraries:

- TkPvm
- TkPen
- TkAudio

They will be described in the following subsections.

2.1.1 Pvm

Whenever any task is subdivided into subtask, several problems arise. For example:

- How to control the creation and destruction of subtask?
- How can separated task run on different machines?

- How can we know that some task is finished?
- How to communicate between tasks?

Basically, these problems are solved by PVM (Parallel Virtual Machine), a public domain software package[1]. Only, PVM is not more than a toolkit with a lot of functions.

In the MIAMI project most software is written using the Tcl/Tk interpreter. This interpreter has a lot of provisions for event-handling, but not for events created by PVM. Also, the only data-type that Tcl knows about are strings. Therefore it was needed to create an interface in which:

- Any time data created by another process arrives, some Tcl-function is executed.
- Data from PVM needs to be converted to strings, the only data-type that Tcl knows.
- Data that must be sent by a Tcl-application must be should be converted to the type that the receiving application expects.

2.1.2 TkPvm

TkPvm is an extension to Tk which adds PVM-commands to the Tk-toolkit. These new commands allow processes to be started and data to be exchanged between processes. The new commands are:

pvm bind `tid msgtag`	bind a Tk-function to the receipt of an expected data-packet
pvm spawn `filename`	to start a PVM process
pvm kill `tid`	to kill a PVM process
pvm send `tid msgtag data`	to send data to another PVM process
pvm recv `format`	to receive data from another PVM process

Software Description

Name:	Tkpvm.c
Language:	C
Type:	Library
Platforms:	(many), PVM

2.1.3 TkPen

TkPen is a Tk-extension which reads data from a pen-tablet. A separate process (pend3) reads the tablet and sends the data to other processes. The TkPvm extension is used to receive the samples in a Tk-application.

Commands:

pen start	start sampling
pen stop	stop sampling
pen `x y p`	move cursor to coordinate x,y with pressure p

Because the pen button is not always handled correctly by Tcl/Tk, a small patch to the original software was necessary. This patch is developed such that it does not disturb any other original functionality if the pen is not used. Also, TkPen makes full use of TkPvm, such that other software parts which receive data are not disturbed by data that comes from this pen extension.

Software Description

Name:	Tkpen.c
Language:	C
Type:	Library
Platforms:	Unix/X11, Mwish, PVM

2.1.4 TkAudio

TkAudio is an extension to Tk, which adds commands for starting and manipulating the audio renderer from the mwish shell. The audio renderer is an independent programm (using PVM) controlling all audio related tasks. The audio rendering software will be explained in chapter 5. TkAudio only provides an interface to the renderer and does not contain any signal processing routines.

Commands:

audio activate `channel`	activate the specified channel
audio init `audio_renderer`	start the specified renderer (see chapter 5 for available renderers)
audio pause	set DAC to pause mode
audio silent `channel`	set channel to silent mode
audio source `filename channel`	sets the specified channel to the file as source
audio stop	stop the audio rendering
audio `elevation azimuth channel`	set the spatialization filter of channel to the coordinates (relativ to head position)
Future extensions:	
audio volume `level channel`	changes the volume of channel

The audio renderer receives the stream of data from different sources (e.g. files, pseudo-devices, TCP/IP sockets), filters the signals for spatial presentation, mixes all sources and sends the data to the digital-analog converter of the machine.

Software Description

Name:	TkAudio.c
Language:	C
Type:	Library
Platforms:	Unix/X11, Mwish, PVM

Chapter 3

Two-dimensional graphics and widget control

3.1 TkPaper

The library TkPaper is a paper-widget for Tk. It functional but still under development. This widget provides a writing surface for handwriting and drawing. It circumvents the standard Tcl/Tk process of drawing on a canvas. The internal procedure is based on checking the difference between a current internal display list and the actual visible bitmap. Continuous drawing of line vectors at a typical rate of 100 XY coordinates per second would be too expensive. For each new coordinate, the complete internal display list is checked. Instead, in the TkPaper widget drawing is done in X11 immediately, and vectors are added to Tcl/Tk without waiting for 'idle' time.

In fact, the TkPaper widget is a mixture of components from the frame and canvas widgets in standard Tcl/Tk. The TkPaper widget is nicely separated from device and communications through the Tcl/Tk binding mechanism. Events from any source (i.e, pointing device) may be bound to the TkPaper widget.

Software Description

Name:	TkPaper.c
Language:	C
Type:	Library
Platforms:	Unix/X11, Mwish

3.2 Mwish

Mwish is a Tcl/Tk-interpreter with the MIAMI-extended commands available. It is the standard "wish" application (the Tcl/Tk-interpreter) with TkPvm, TkPen, TkAudio (from Bochum) linked in. As a consequence, with Mwish (as opposed to plain wish) it is possible to incorporate external events. This is done by use of the PVM package. In addition, a dedicated library is used for pen input, which uses the PVM socket communication. Finally, Mwish allows for audio output control. In the future more extensions needed for the MIAMI-project will be linked into Mwish.

Software Description

Name:	mwish package
Language:	C
Type:	Main program
Platforms:	Unix/X11, PVM

Chapter 4

Three-dimensional graphics and widget control

4.1 Multi-dimensional control through V-scope

Our distributed architecture is based on Unix (SGI Indigo and Sun Sparc) and Win32 (486 and Pentium) platforms, with particular regard to Windows 95. Figure 1 depicts the distributed architecture developed for the experiments in WT2.3. We developed a library based on sockets - both under Unix and under Win32 - which allows the communication between processes possibly running on different machines. A software module for the integration with the PVM distributed environment is in course of development. The software and hardware described here forms the basis for the Exoskeleton application to be reported later.

4.2 The HARP/V-scope experimental software

The HARP/V-scope experiments have been implemented in both a distributed environment and a single workstation. In the first case the experiment has been implemented in two workstations running the beta version (currently, beta 3 build 347) of the Microsoft Windows 95 operating system. The V-scope sensor system hardware is linked via a high-speed serial interface (RS232C with 16550AF UART). In the local implementation, we use a single Pentium 90 machine (32MB RAM, an MGA Impression+ video board, and a Sound Blaster AWE32 sound board), under Windows 95, in which all the software runs locally. In both cases we use the same external audio hardware, as shown in figure 1.

The HARP/V-scope application developed for experimenting sound/movement interac-

tion is composed by four main subsystems:

1. (a) input: the V-scope interface;

2. (b) pre-processing and force field metaphor interaction;

3. (c) movement recognition;

4. (d) output: sound and music, computer animation.

4.2.1 Subsystem (a) - V-scope interface

The VScope agent is designed to acquire the information on the position of a number of V-scope markers, typically placed on the body of a user. It manages both the low-level serial communication and the link with client modules. V-scope is a IR/ultrasound sensoring device developed by Lipman Ltd. for the real-time acquisition of the position of up to eight markers placed on the human body (e.g., on the articulatory joints) or in general on moving objects (e.g., a video camera). The hardware is composed by the markers, three tx/rx towers for real-time detection of markers position, and a main processing unit connected via a serial link to a computer. The sampling rate can vary from 5 to several hundreds of milliseconds per marker (20ms per marker is currently used). As for the limits due to the V-scope acquisition hardware, we are able to manage a stage whose dimension can vary from 2 to 5 meters in depth: faster sampling corresponds to a smaller area, due to the limitations of the ultrasound sensoring devices. Our experimental results show that a 12-15ms per marker is the best tradeoff between speed (a good value for human movement acquisition without loosing too much information) and stage size. The precision of the V-scope hardware is in the range of 1cm, acceptable for our application. The Vscope interface consists of an executable and two DLLs (Dynamic Link Libraries): the executable is the user interface manager and provides means for configuring V-scope settings. Low-level methods for the V-scope hardware management are encapsulated in a Microsoft Windows DLL; high-level intermodule communication methods are encapsulated in a Shared DLL (a shared-memory object). Thus we have the running EXE file which communicates with the V-scope hardware via the low-level DLL and stores the data acquired in real-time in the shared DLL, making them available to one or more clients. Besides the VScope agent itself, further agents are available: for example, the VScope Monitor is used to monitor the status of the markers placed on the dancer's body.

4.2.2 Subsystem (b) - Preprocessing

The movement data pre-processing agent can be linked either to local agents (via Microsoft OLE - Object Linking and Embedding) or to remote agents (via our Ethernet/WinSock library). This depends on the global requirements of the application: the HARP development environment is in charge of allocating agents and creating their links in the network. In a distributed environment, the motion data preprocessing agent actually executes three main tasks: it has to manage two local connections (with subsystem (a) and the Force Field Navigator agent) and one remote connection. The link with V-scope, as mentioned earlier, is achieved via a shared DLL, the link with the Force Field Navigator is based on Microsoft OLE Automation and the network link is built on standard Winsock libraries. The raw data stream from V-scope is immediately filtered, to make sure no spurious information are present and values are within a meaningful range.

4.2.3 Subsystem (c) - Movement and Feature Extraction. The force field metaphor

The communication agent reads the sensors data stream and passes it to the agents for gesture recognition and movement analysis, whose output is available to trigger or influence the activities of sound/music and animation agents. As a simple example, a feature agent might recognize that the dancer has raised his/her left arm over a certain threshold, and its output can be used to activate a certain sound processing agent. The gesture extraction task is further subdivided into several concurrent agents, each dedicated to a different kind of movement recognition task, according to the current scenario. The Gesture/Movement agents implemented in this experiments are able to recognize several different features and gestures: raising and lowering one or both hands, raising and lowering the body, opening and closing the palm of both hands, distance between hands and gesture speed. An interesting category of agents for the interpretation of movement data is based on the force field metaphor: for example, in the demo videotape we mapped the (x,y) coordinates of a marker into a force field whose three areas around peaks corresponds to similar areas of the sensorized stage characterized by different behavior (different mappings of movement/sound). The agent continuously reads the field data corresponding to the current (x,y) position and makes it available for further processing. Other agents in course of development will be able to extract higher-level features and gestures from the movement, to model complex music/movement correlations. Examples of high-level features are "how fast the movement is", "how a tempo the dancer moves". This is a kind of information which is the result of the integration over a time window. Following

the results of the research in auditory perception (Leman 1990), two different ranges of time window, approximately 0,5-1s and 3-5s. are used. Experiments are in progress based on self-organizing neural networks for the classification of incoming data from sensors, including acoustic signals.

4.2.4 Subsystem (d) - Output Generation

In the simplest case, the recognized features can be linked to events: the system is presently designed to control sound and music in real time. A next step will regard the control of computer animation. The mapping of the performer's movement to sound and animation agents can be either pre-defined or dynamically updated according to the information acquiired by particular feature extraction agents. This last case currently under development, based on the new HARP model. The MIDI standard is used for sound event control: sound output agents receive MIDI commands through OLE links from movement recognition agents, and enqueue them on the MidiKer agent, which manages the low level scheduling and synchronization and the output to synthesizers.

Chapter 5

Audio rendering software

Chapter 6

Specific Experimental Applications

6.1 Simple 2D Demo

A demo "2d-demo" is written to test the possibilities for Tcl/Tk to do real-time multimodal input. On screen several two-dimensional objects (squares and circles) can be created, selected and deleted. Using the mouse, these objects can be moved and resized directly in real-time. Casual experimentation revealed that the system response is virtually immediate in systems varying from Sparcstation-2 or VAXstation 3100 (relatively slow systems) to Sparcstation-20 or HP 9000/735 (fast systems).

Software Description

Name:	2d-demo.tcl
Language:	Tcl/Tk
Type:	Main program
Platforms:	Unix/X11, wish

6.2 Multimodal Editor

A multi-modal editor, which can handle text, images, sound and handwriting. It is written using the Mwish interpreter. You can load, edit and save the following file-formats:

- GIF, XPM, PPM (images)
- ink (.ink)
- audio (.au)
- text
- html (combination of all of above formats)

If you click on any object (text/image/sound) using the right menu-button a menu appears in which you can change the displayed format. For example you can change text to sound (by a speech synthesizer). Or handwriting to text (by a handwriting recognizer, which is not implemented yet).

This editor is merely a framework in which different modalities can co-exist in a single document, and conversions are fully controlled by the user.

Software Description

Name:	editor.tcl
Language:	Tcl/Tk
Type:	Main program
Platforms:	Unix/X11, (M)wish

6.3 Mdraw

A small drawing-editor using the TkPen extension. The user can draw with the pen. Store the handwriting in a file and load it or play it in real-time.

Software Description

Name:	mdraw.tcl
Language:	Tcl/Tk
Type:	Main program
Platforms:	Unix/X11, Mwish

6.4 Virtual Keyboard

A virtual keyboard, to allow pen-users to type characters by clicking on a keyboard which is graphically displayed on the screen.

Software Description

Name:	vkb.tcl
Language:	Tcl/Tk
Type:	Main program
Platforms:	Unix/X11, Mwish

6.5 The Exoskeleton system

This experimental hardware and software system developed deals with the real-time animation of a human model based on the movements of a real human, acquired by a sort of exoskeleton system.

6.5.1 Hardware

The main characteristic of the exoskeleton we have built is "modularity":

- Every joint is a simple rotational joint without joint limits;

- Composite rotations can be realized assembling different base units (see figure), as in a sort of "Lego"-like system;
- Each joint is based on a high precision potentiometer, connected to a 16 bit AD input;
- The complete structure can be worn by the actor, who can freely move on the stage without the limitations described for the CosTel system.

6.5.2 Software

The acquisition rate is about 1 Khz, so the joint measures can be used to reconstruct real movements. Data acquired on the PC are send via sockets to the graphical workstation (SGI Indigo), where are used to animate in real time the model. "Alice", this is the name of the graphical animation, has been developed using the SGI Inventor Toolkit, a graphical tool extremely efficient to realize and real-time control complex kinematic models. The refresh time we obtain is about 20 hz, that is not the optimum, but enough to have a real tracking of the actor movements. The joint angles are also sent to a second workstation for music generation, that in real-time modifies the main parameters (pitch, modulation, timbre and amplitude) of the sound synthesized.

6.5.3 Experimental application

One of the first experiments made with this architecture has been the "virtual cello player" where the demonstrator used the motions of his hands to "emulate" a cello player. This is shown in the DIST Miami demo videotape. The future activities involved in this project are toward the realization of an infra-red link between the skeleton and the personal computer to obtain a wireless system, the optimization of the mechanical structure, and on more sophisticated sound/music integration.

Bibliography

[1] J. Kowalik, editor. *PVM: Parallel Virtual Machine: A Users' Guide and Tutorial for Networked Parallel Computing*. MIT-Press, Scientific and Engineering Computation, 1994.

[2] N. Mayer. XWebster: Webster's 7th Collegiate Dictionary, Copyright ©1963 by Merriam-Webster, Inc. On-line access via Internet, 1963.

[3] J. Ousterhout. *Tcl and the Tk Toolkit*. New York: Addison-Wesley, 1994.