ESPRIT '84: Status Report of Ongoing Work
J. Roukens and J.F. Renuart (Editors)
Elsevier Science Publishers B.V. (North-Holland)
© Commission of the European Communities, 1985

INTELLIGENT WORKSTATION IN THE OFFICE
State of the Art and future perspectives

N. NAFFAH - BULL TRANSAC - France

G. KEMPEN - Katholieke Universiteit Nijmegen - Netherlands

J. ROHMER - BULL Centre de Recherche - France

L. STEELS - Vrije Universiteit Brussel - Belgium

D. TSICHRITZIS - Crete Research Centre - Greece

G. WHITE - INRIA - France

The Intelligent Workstation project aims at the implementation of a complete office system including workstations and servers interconnected through a local network. The workstation itself will have a powerful architecture based on standard VLSI chips and will provided a highly interactive interface.
On top of its distributed operating system, an office information system will be designed helping the user to realize his tasks. Natural language processing will be used for document elaboration and an audio package will provide a powerful multimedia interface. On servers, dedicated hardware such as the SCHUSS filter will accelerate the access to large knowledge bases.

INTRODUCTION

The Intelligent Workstation will have to be supported by and give support to a variety of information transfer and storage facilities. These include such things as file and object servers, local area networks, the telephone network, commercial large scale computer networks and other specialized devices for data Capture and manipulation.

Experience gathered with the buroviseur, created by the Kayak group, and observation of other systems now being marketed in world markets has been used to formulate functional specifications for the next generation of office workstation. It is evident that the system will have to support a wide spectrum of tasks being performed in contemporary offices and will therefore have to have a wide spectrum of facilities such as i/o devices, speed, memory and the like.

This, in turn, requires certain specific engineering specifications which, in the large, can be listed as follows :

- a powerful CPU which is capable of using a large logical address space and high speed RAM chips and which can support a high resolution screen.

- a large random access memory which is not too expensive but which can reduce the number of relatively slow accesses to secondary storage.

- a high resolution, not too expensive, bit mapped display screen.

- a variety of i/o devices such as mouse, speech recognizer, speech and sound generator as well as a well designed, ergonomic keyboard.

- a high-speed local area network interface.

To realize these requirements and at the same time to produce a product which is competitive in world markets it is desireable to integrate as many circuits as possible and place all the circuits required on a single board. Usage of VLSI for accelerating the execution of some functions (e.g. rasterop) is envisaged. Those devices which are considered optional can be located on standarized plug-in boards.

Some of these aspects are discussed further in the following paragraphs.

THE PROCESSOR

The processor must offer the best ratio of price to performance when all factors are taken into account. These include the cost and complexity of the other supporting chips, the availability and possibilities of second-sourcing everything required to support this chip and the cost of producing software. At this time, the available candidates are the INTEL 80286 (16 bits), the MOTOROLA 68020 (16/32bits) and the NS 32016 (16/32 bits).

The 80286 driven by a 8 MHZ clock is equivalent to a 16 MHz chip since a minor cycle uses 2 clock pulses rather than 4. Of the three chips, this is the most powerful. Unfortunately, to exploit this power, other complex circuitry is required, interlaced memory, a special memory controler chip, the 8207 and very fast memory (100 ns) which is presently very expensive. Furthermore its internal architecture is relatively weak (4 data registers and 2 index registers) and memory is logically segmented into 64 kbyte blocks. This implies instruction sequences which are more complex and longer than would otherwise be necessary.

The 68010 and 32016 can be driven at 11 MHz (4 clocks per minor cycle) without wait states, using the more common, 120 ns RAM chips and without interlaced memory. The two have a similar architecture, however the 68010 has more registers, more addressing modes and an accelerated "loop" mode. There are several second sources for the 68010, including an european company (Thomson, beginning 1st quarter 1985) and there are presently complete evaluation cards and several operational systems. A 68010 driven at 11 MHz is equivalent to a 0.8 or 0.9 Mip machine or about 4 times faster than the present buroviseur. The code developped on the 68010 can be run also on the downwardly compatible 68020 (32 bits) when it appears in 1985.

The 68020 has therefore been chosen as the target chip of choice. This should not be considered as absolutely final as things change very rapidly in this business.

THE SCREEN

At the present time, there is a significant price break for screens having a sweep rate of about 30 kHz. Below this value screens are available for about 150 Ecu. To exploit this advantage, it is proposed to use a screen with a resolution limited to about 600 x 800 pixels requiring about 64 kbytes of bit-mapped memory. It is, however, perfectly possible to use a higher resolution screen with a resolution of 1024 x 768 requiring a memory of 96 kbytes for a cost of about 600 Ecu. The size of the screen will be 15 inches in diagonal measurement and be capable of generating from 50 to 60 images per second.

THE MEMORY

Experience with the buroviseur has demonstrated the necessity of a large memory, at least 1 megabyte. Such a memory can be realized with 32 memory chips of the presently available 256 K x 1 bit configuration. 4 additional chips can be provided by replication. For simplicity, the memory will be refreshed directly by the CPU thus eliminating a specialized memory controller. This task will occupy the CPU for 80 us every 2 ms, effectively wasting 2,5 % of the available cycles. The memory will be able to function at 11 MHz without wait states with 120 ns RAM and if necessary, at 12.5 MHz (1 Mip) with 100 ns RAM.

The screen memory will be implemented with special chips, 64 or 126 Kbytes of Texas Inst. 4161 i.e. 8 or 16 chips.

The 68010 can manage virtual memory. The memory management controllers presently available require the wasting of 1 cycle per access, therefore we propose to use a translation table resident in RAM with a paging mechanism operation on blocks of 4 kbytes. This system permits the efficient use of the entire memory space seen as 1 continuous block, even if it is partly discontinuous. The architecture of the system will permit the physical memory to achieve a size of 8 Mbytes.

THE LOCAL AREA NETWORK

The local area network will be of the ETHERNET type using available chips compatible with the selected CPU. This permits the transfer of data at a maximum rate of 10 Mbps. To reduce costs, a version of this system called "Cheapernet" is proposed. This version uses cheaper cables and standard BNC connectors to be used and avoids the use of the expensive couplers required by the XEROX system.

THE DISKS

We propose to use one integrated circuit which controls both a floppy disk and a Winchester (with a minimum of 10 Mbytes capacity), compatible with the CPU, which can also control the DMA (thus there is no need of a special controller). Presently there are 10 Mbytes Winchester disks available with average access time of 50 ms, about half that of the buroviseur disks. Other types of disks, such as numerical optical disks may be used by some of the workstations, according to their specific requirements.

I/O MANAGEMENT

For greater processor efficiency, we propose to use a processor to handle the keyboard and the mouse.

TELEPHONE INTERFACE

It will be based on new available circuits.

AUDIO HARDWARE

Specialized devices are to be used for the recognition and generation ef human speech and possibility for the sampling and compression of audio signals.

EXTENSION BUS

To facilitate the connection of peripheral devices to the buroviseur, a standard bus (e.g. VME) may be used.

THE SCHUSS FILTER

Here we describe the basic principles, the architecture and some possible applications of a processor called the SCHUSS filter. The SCHUSS filter can be seen as a device with two inputs and one output ; the first input is the filtering criterion (a program) ; the second input is the data to be filtered (in a sequential way). The output is the data which fulfills the filter criterion. Under the architectural point of view the SCHUSS processor can be as specialized processor (in filtering) but also as a general purpose processor. Working as a specialized processor, it can process "on the fly" data coming from a disk where "on the fly" means at the normal disk transfer rate. Even if disks transfer rates are now up to 3 Mbyte/sec, SCHUSS can execute searches of reasonable complexity in one pass. The idea is to off-load the host processor of search tasks and other kinds of processing.

The SCHUSS filter was designed to work with the Buroviseur ;

The strategy to filter data is to use automate technique as described in (Rohmer 80 and Rohmer 81) The basic idea is to consider sequential data to be processed (files, messages) as a language and queries as a grammar. This technique is very general and allows treatment of complex data structures (flat, hierarchical and texts as well as numerical values).

The SCHUSS filter is a processor. Externally it has two busses, one to access a memory and an I/O space (16 bits data 16 bits address), and the second one to get data to be filtered (8 bit data and 10 bit address).

Internally it is a microprogrammable processor. Many of the possibilities of adaptation to different data formats come from this feature. The microinstructions have a horizontal format of 112 bit, this allows gains in speed, making treatments in parallel in the two ALUS.

An instruction set is microprogrammed. For each new application, this set will be enriched to deal with specific problems.

The filter installed in the Buroviseur consists of two boards

- a processor board
- memory and interface board

The operating system in the Buroviseur is modified in order to offer the necessary entry points to establish the transfers between the host and the SCHUSS processor.

A typical application for the SCHUSS filter is to accelerate the response time in a data base.

Four steps are needed for a filter operation in this application.

- a query is compiled the host, the target language is the instruction set of SCHUSS tailored for this application.

- compilation results are loaded into the filter memory. This is the filter criterium.

- the filter executes this program, processing data coming from a disk 'on the fly". Only one pass is needed.

- the filter sends back the results of the filter operation to the host memory

The response time for a given query is almost the time to read the file from the disk. The time for compilation and communications is very short.

Others applications are described in (GONZALEZ - 84)

THE OPERATING SYSTEM

Basic software develoment

Basic software for the multimedia workstation should have state of the art distributed systems and artificial intelligence support. The problem is to combine both things in a single kernel.

Our experience leads us to :

- port an existing distributed system : CHORUS
- modify and extend it to this project requirement

This allows simultaneous work, as described below.

Preliminary studies

These studies must define the underlying software required by an artificial intelligence system :

- which features could be supported at the application level ?

- which ones should be implemented at the kernel level ?

Some of them may be found in existing expert systems but distribution introduces extra complexity and needs : distributed synchronization etc...

For all these reasons these studies will be undertaken at the very beginning of the project, in parallel with the first implementation of CHORUS :

Implementation

1) Transport and adaptation of an existing distributed kernel (CHORUS) including :

   . Installation of a developement environment (Software Factory) compilers and assembler, linker, source code and documentation management tools.

   . Adaptation of CHORUS to the new hardware (memory and CPU management, basic controllers, networking devices).

   . Debugging and bench marking of CHORUS.

2) Adaptation, extension and optimization in accordance with supported software characteristics :

   a. Real time mechanisms ; Resource allocation strategy
   b. support of Multi Media devices : specific handlers, filter
   c. Introduction of object management primitives and distributed synchronization functions

Delivery of a configuration and generation toolkit for further extension

Part 2 may vary, depending on other participants'experience and specifications (mainly point c.). It may also include some "system applications" as maintenance tools. That is why a two step transportation was choosen in order to provide as soon as possible a reliable operating system kernel. Part 2 realisation could therefore be influenced by first user's experience.

THE INTELLIGENT OFFICE SYSTEM

The goal of this module is to demonstrate the feasibility of a knowledge based Office System. Such a System can be used to embed knowledge about office processes and office problem solving activities. Thus, the office workstation becomes a truly powerful tool in the performance of office work.

In the first year of the project, the work concentrated on the development of an efficient reasoning mechanisms for using this knowledge. In the following years of the project, the focus of research will switch from knowledge representation to the use of a knowledge Representations Systems (KRS) in an office environment : An Intelligent Office Information System will be built given the tools and features of the Knowledge Representation System designed and implemented in the first year.

Building an Office Information System involves 3 tasks :

1) The analysis and implementation of the structure of office into a working model

2) The construction of a user model so that novices can use and change these structures

3) The integration into an actual office, including connections to printers, mailsystem, etc.

The first task includes the study of Office Semantics in order to develop a technology for the proper representation of an office and the construction of expert decision aids. A model of the office will be constructed, which will be capable of representing and reasoning about tasks and procedures in the office. Not only documents and forms but also concepts such as tasks, goals, procedures, plans office workers, their roles and responsabilities will be represented. This konowledge about the office will be structured in an open-ended knowledge base, which will allow the office workers to access and process information easily and will helps them in understanding their work environment better. The description of office tasks in terms of explicit goals and hierarchical planning trees will make it possible to computerize office procedures in a very flexible way. The system will primarity play a role in supporting the planning being done by the user by helping represent, manage and communicate the resulting plans.
Once a satisfying model for Office Semantics will be found, an Integrated Office System will be constructed. This OIS will use the representation of the office in order to support various knowledge intensive office activities :

- it serves as an information source by providing a descriptive framework in which to express the specifications of what tasks are to be done, who is responsible for doing them, and how they are to be done,

- it provides a terminology of describing office work in uniform way,

- it helps in analysing and monitoring the carrying out of tasks, tracking the process of execution and does some of the procedural steps itself,

- it provides automatic planning and problem solving to determine what actions must be taken to acomplish a given goal,

- it serves a communication device for supporting interaction between people carrying out large, interdependent tasks

- it assists the manager in organising and scheduling the work within the limits of time, and the constraints of manpower and budget,

- it allows the office worker to access and process the computerised data in a very flexible way.

Ideally, the office system is totally integrated in all the functions of the office : all flow and processing of information takes place through the system. The system constantly looks at the activity taking place and takes the initiative of cooperating at every possibility.

The second task involves the design and construction of a user-model of the Office Information System. This model will shadow the underlying implementation and will provide a simple and natural interface.

Very often users complain that interactive office systems are not flexible enough : flows in the initial design are difficult to correct, and when new demands from the users arise during the systems lifecycle, they are supported too late or not at all. Especially in an office environment, which is constantly evolving and changing, the office workers should be provided with the ability and the tools for programming their workstations. We plan to support the office worker in this communication with the Office Information System by placing a high level, intelligent user interface at his disposal. For the construction of this 'communications language' several approaches will be investigated :

- Programming by example is one the methodologies that has been proposed for supporting the development of office applications by non computer scientists. Programs are built by performing direct manipulations on visually presented on display, simulating the execution of the program on exemplary data items. Several systems have been proposed that provide programming facilities by means of examples : SBA DeJong,Zloof,77), TINKER (Lieberman, Hewitt, 80), and others. These different approaches will be studied and elaborated in the context of a knowledge-based office system.

- Programming by descriptions
  In this case the conversation between the user and the system occurs on a higher level : it is based on semantic terms. This means that objects are identified in terms of their attributes, properties or relationship with other objects. The communication occurs by means of a description language such as OMEGA (Attardi,81) or XPRT (Steels,79).

- Graphical programming
  This means programming language which have a graphical syntax. E.g. programming with boxes (Cardelli,82) or (Knuth,79) or (Lakin,80), the use of successive menus to guide users etc.

Concerning the last part of the project, our objective is to do a complete case-duty. An actual office will be studied. A Knowledge Based Office System will be designed, built and integrated into that office. Concrete experiments will be performed not only to see whether it is technically feasible to implement office knowledge, but also to test the interface with novice users and to see if there are any gains in the productivity and the effectiveness of the office.

NATURAL LANGUAGE PROCESSING

Desired functionality of the Author System

This section outlines the new functions that we plan to build into the author system in the course of the project. This first example deals with text modifications at the work level. If you want to pluralize a noun which occurs frequently in a text, then current wordprocessors allow changing this noun to its plural form by issuing a single command. However, the linguistic changes which this modification entails with respect to other parts of sentences cannot be computed automatically. For example, "this document" would have to change into "these documents". Also in each sentence where this pluralized noun phrase plays the role of grammatical subject, finite verbs need to be pluralized as well in order to maintain subject-verb agreement.

Furthermore, all pronominal references ("it") to the plural noun have to be replaced ("they" or "them", depending on syntactic function). In current wordprocessors this is done by hand. Especially in languages having a rich morphology like French, German and Dutch this is a time consuming and error-prone process. In an author system as we envisage such changes propagate automatically.

At the sentence level, an author wishes simple means of altering the order of words and phrases in accordance with rules of the language (cf. the differing obligatory word orders in main and subordinate clauses in German and Dutch), for active/passive transformation, for adding coordinating constituents, and for suggesting paraphrases. Such facilities will necessitate a sophisticated user interface which enables easy access to the linguistic structure of each sentence. For instance, writers should have the possibility of viewing "syntactic trees" on a high-resolution screen and to address not only letters, words and lines as in current wordprocessors, but also syntactic units such as parts of speech, noun phrases, clauses, etc.

Author systems will be particulary useful at the discourse level . A relatively simple application is the automatic generation of large numbers of individualized documents (e.g. business letters) which need more linguistic variation than can be handled by current menu-based report generators using templates. Consider the following template : Following your order "invoice" which was placed on "date" we have sent you number" copies of "itemname". Now suppose a costumer has sent two invoices that came in on two consecutive days. The expanded form needed in this case could be a sentence like

Following your orders ZO12145 and ZO1246 placed on May 12 and 13 respectivily, we have sent you 10 copies of Manual A and 1 copy of Manual B.

In the author System, we envisage this amount of linguistic variation can be handled easily. More difficult applications concern the fully automatic composition of non-standard texts on the basis of a set of data. A recent example is provided by a project of Bell Laboratories and Carnegie-Mellon University. Kukich (1983) wrote a program which analyzes trends óf share prices at the New York Stock Exchange and reports these in the form of written texts.

At the discourse level, our goals will be very modest. We do not aim at anything more sophisticated than generating natural souding descriptions of KRS objects which are fully specified in a databank. An example in the area of document tracking might be the automatic generation of reports listing the history and current status of documents processed within the office. In this manner we can avoid complex reasoning about "world knowledge".

Integration of Dialogue and Author Systems

The linguistic module and author systems overlap to a considerable extent. For instance, both systems need quick access to an on-line lexicon and a grammar as well as to word and sentence parsers and generators. In the accompanying figure we therefore present the rough design for an integrated-dialogue and author system as part of an office workstations's software tools. Through the dialogue components, the user can consult domain knowledge which is relevant to the goals and procedures of the office. One such domain concerns stored text documents which have been processed (written, edited, criticized, authorized, mailed, received, translated, updated, etc.) by office workers. A typical user question might be : "Did I finish and mail the letter to Mrs.X ?

After having located and retrieved the document, the dialogue system hands it over to the author system, which proceeds by displaying the text in a window on a screen.

An important assumption underlying our design is that the author system can treat a text as a sequence of characters (stored in a file, displayed on the screen) and as a linguistically structured object. Both representations ("orthographic" and "linguistic") of texts are maintained an operated upon interdependently. For example suppose the user instructs the author system to put a passive sentence into active voice (by late linguistic structures, in particular syntactic trees). The author system then responds by modifying not only the linguistic structure of the sentence directly, e.g. by inserting the plural ending of a noun, then the author system will automatically adapt the corresponding linguistic structure and carry out any implied alterations to other parts of the sentence.

Thus users have two ways of editing a text. They can directly modify its orthographic representation by typing into a diplayed text file. (This is the procedure followed in present-day word processors). In addition they can call the Tree Editor and propose alteractions to the underlying structural (linguistic) representation. In the former case, the parser components will take care of adapting the corresponding linguistic structures , in the latter case, the word and sentence generator will reconfigure the linguistic structure. Both procedures will cause text file and screen image to contain not only the writer's explicit edits but also the ones that are entailed on linguistic grounds.

Components tasks

The work leading to implementation of the system described in the previous section can be divided into the following six component tasks

A) Specification of the linguistic knowledge (Dutch morphology, syntax and semantics, context knowledge needed by the linguistic modules ;

B) Implementing flexible and robust linguistic processors (parsers and generators at the level of morphology, syntax and semantics) which can recognize and correct grammatically deviant input.

C) Implementing the Author System (i.e., the interfaces between user, Tree Editor and Text Editor), and the editors themselves

D) Implementing the Dialogue System (i.e the interfaces between user and linguistic processors) for natural-language interaction with intelligent office application provided by other contractors ;

E) Transport of a complete system to workstation built in the project ;

F) Representing and programming the linguistic knowledge for the second target language (French of English).

The partner in charge of this subtask already spent in the order of 25 man years on tasks A through D. On the basis of what we have accomplished in that amount of time, and considering that we will work in advanced LISP programming environment (Symbolics Lisp Machine, we estimate that 10 additional manyears will be minimally necessary to complete tasks A through F, i.e., to put a prototype of a dialogue/author system on the new workstatin. We presuppose that several partners help in designing and testing the user interfaces for author and dialogue systems, and that transport of software from Symbolics/VAX machines to the workstation will be relatively easy.

MULTIMEDIA INTERFACES

A preliminary step in the implementation of a multimedia interface is to augment the programming environment of the prototype workstation with mechanisms that give flexible software control over the devices attached to the workstation. We assume an environment base on object oriented languages (i.e, classes and methods), a programming methodology known for its flexibility and ease in implementing prototypes. Windowing packages that support the screen keyboard and mouse are now available on many workstations. There is no support though for the remaining devices we expect to be connected to prototype multimedia workstation. In particular audio devices have received little attention. Thus we have begun the design of an "audio package", analogous to the present windowing and graphics package, for handling the audio devices.

To describe the audio package in more detail it is necessary to describe the various representations for audio information and the hardware we wish to support. Concerning the problem of representation, we will assume that two formats are used. The first format is simple digitization of the audio waveform leading to data rates of the order of 64 kilobits per second. The second format uses linear predictive coding (LPC) to compress speech waveforms to a few kilobits per second. We will call these two representations <u>audio 1</u> and <u>audio 2</u>

The difficulty with audio data is that there are often strict real-time constraints. For example, when playing back stored voice one must take care to keep the digital-to-analog converter supplied with data from the disk. Thus we have chosen to base the audio package on groups of realtime processes that are activated and "configured" (this will be explained shortly) by a parent process and then executed concurrently.

There are two objects classes found in object-oriented programming environments that are useful for constructing the audio package. These are processes and streams. Our approach is to include the workstation's audio devices as part of the programming environment by representing them with special objects that produce and consume streams. In general these objects will deal with streams of audio1 and audio2, however we will also use text (i.e. character) streams and streams of uninterpreted digital data. In addition, a set of real-time processes are used to aid in linking (i.e., configuring) the streams used by these objects. The special objects and processes are :

<u>Special Objects</u>

analog-to-digital converter	- produces an audio1 stream
digital-to-analog converter	- consumes an audio1 stream
LPC analyzer	- consumes an audio1 stream
	- produces an audio2 stream
LPC synthesizer	- consumes an audio2 stream
	- produces an audio1 stream
voice recognizer	- consumes an audio1 stream
	- produces a character stream
voice synthesizer	- consumes a character stream
	- produces an audio1 stream
digital telephone	- consumes a digital stream
	- produces a digital stream

Processes

diskwriter	- consumes a digital stream
diskreader	- produces a digital stream
tee	- consumes a digital stream
	- produces two digital streams
	- consumes two audiol streams
mix	- produces an audiol system

The above objects (processes are also objects) respond to messages which control their state and behavior. The general operation of these objects should be apparent from their name with the possible exception of the tee and mix processes. A tee process is very simple - it merely duplicates its input on two output streams. A mix process superimposes two audiol streams (this may not be feasible for audio 2 streams).

The audio package gives the interface implementor the ability to activate and interconnect the above objects. For example, a telephone conversation requires using the output stream of the analog-to-digital converter as the input stream to the telephone and using the output stream of the telephone as the input stream of the digital-to-analog converter. If compression is used the LPC devices can be added to the "circuit", to store the conversation one uses the tee and diskwriter processes, and so on.

In the long term we plan to develop tools for interface design and implementation and so avoid the ad hoc addition of the user interface on top of existing hardware. A collection of such tools is commonly referred to as a user interface management system (UIMS).

The choice of high-level buiding blocks for the interface has centered on templates. A template is an encapsulation mechanism for interface objects (menus, windows, icons etc ;) - the information appearing within a template definition includes, the formating environment within the template, the system events that template responds to, and the actions invoked by the template. As a specific example, suppose one decided on, as part of the user interface, a graphic symbol to represent the station's mailtray. The template definition would then describe the form of the symbol and indicate that it is associated with the mailtray. If one wanted the symbol to change colour or intensity, or to be altered in some other manner when mail is received, then this information would be added to the template definition. Note that it is the template definition that is altered rather than the mail system. This example illustrates how templates divorce the user interface from the functional components of the office system.

Templates are medium specific. There are audio templates for audio devices, images templates for image devices and text templates for character devices. A particular data structure may be displayed in a variety of templates, not neccessarily all for the same medium. For example displaying a text letter in a text template one can only use the standard ascii character set. Using an image template one would have access to arbitrary soft fonts.

One of the most important issues regarding a user interface management system is to fully utilize the workstation's i/o devices. Thus work carried out on the multimedia interface will require identifying a rich set of template operations that involve the station's audio and image devices. A small set of powerful template operations has already been proposed. These operations (known as the embed, present, assign, and extract operations) allow high-level manipulation of structured audio, image and text data values. For instance, extracting text from an audio template corresponds to speech recognition, while embedding text in such a template would involve speech synthesis, representing the template corresponds to speech output and assigning the template to speech storage.

Once the user interface management system is available, providing the system implementor with a flexible tool for experimenting with the user interface, it will be possible to explore more fully the interplay between sound and images (and other forms of data) and their affect on the dynamics of user interaction.

-o-o-o-

REFERENCES

(Attardi, 81) Attardi G. and Simi M., Consistency and completeness of OMEGA, a logic for Knowledge Representation.
Seventh International Joint Conference on Artificial Intelligence, Vancouver, 1981

(Cardelli,82) Cardelli L., "Two dimensional syntax for Functional Languages", Proceeding of the European Conference on Integrated Interactive Computing Systems, 1982 Stresa, Italy, 1982

(DeJong, Zloof, 77) DeJong P., Zloof M., "The System for Business Automation (SBA) : A Programming Language", Communications of the ACM, Vol. 20 No 6 (June 1977), pp. 385-396

(Gonzalez 84) Gonzales Rubio R., Rohmer J. and Terral D.
The SCHUSS filter : a processor for Non-Numerical Data Processing, 11th Ann. Symp. on computer architecture ann arbor 1984,

(Knuth, 79) Knuth, D 79 TEX and Metafont, Digital Press.

(Lakin, 80) Lakin F., "Computing with text-graphic forms"
Proceeding Lisp-conference, Stanford, 1980

(Lieberman, Hewitt, 80) Lieberman H., Hewitt C., A Session with TINKER : "Interleaving program testing with program writing", Conference Record for the 1980 LISP-Conference, Stanford, 1980.

Rohmer J. Machines et langages pour traiter les ensembles de données (textes tableaux et fichiers). Thèse d'Etat. Université de Grenoble 1980.

Rohmer J. Associative filtering by Automata a Key operator for Database Machines. Proc 6th Workshop on Comp. Arch. for Non-Numerial Processing; Hyers. 1981

(Steels,79) Steels L., "Reasoning modeled as a Society of Communicating Experts", M.I.T., Artificial Intelligence Lab. TR 542, June 1979