Architectural Aspects of Multimedia CD-I Integration in UNIX / X-Windows Workstations

Dr. Klaus Meißner

Philips Kommunikations Industrie
PoINT Software & Systems
D-5900 Siegen
Germany
Phone : (+49) 271 / 380 - 2574
E-Mail: meissner@pki-si.philips.de

Abstract:

This paper describes the integration of CD-I systems in UNIX and X-Windows/Motif workstations. It is based on results of the ESPRIT project MULTIWORKS. First a short overview of the main CD-I features is given. A description of the basic requirements for the integration of CD-I in professional workstations follows. One of the major requirements is the support of off-the-shelf CD-I titles running in a Motif window concurrent with other UNIX applications and the support of new UNIX/Motif multimedia applications, which use the CD-I player as a digital video, sound and CD-ROM sub-system. The client / server architecture and the separation and distribution of time critical multimedia API functions over both systems (UNIX WS and CD-I system) are presented in more detail with special emphasis on the critical problems concerning real-time handling. Finally some experiences with the system are described.

Keywords:

Multimedia, Optical Storages, CD technology, CD-I, DVO, Multimedia API, UNIX, X-Windows, Motif

1. Introduction

New market research about trends in professional multimedia solutions [1, 2] shows that CD based multimedia business and kiosk (point of information / sales) applications and systems will be the fastest growing segments in this market. The precondition for this commercial success is the availability of at least de-facto standards on media, data format, system and application programming interface levels and also the availability of sophisticated (authoring) tools supporting these standards. For CDs several world wide (de-facto) standards are in place [2], including those for the physical level (IEC 908, ISO 10149), the logical level (ISO 9660, CD-ROM XA = Yellow Book, CD-RDx), for data formats (CD-ROM XA, MPEG, JPEG), for the API level (X/Open XCDR, CD-I API) and for system level (Green Book = CD-I Specification). With the Multimedia PC (MPC) initiative and the release of Microsoft's Multimedia Windows another de-facto standard on system level for PCs will be established soon. This standard will support CD-ROM XA technology. All these platforms including the Macintosh and NeXt workstation have different multimedia characteristics and currently different multimedia application programming interfaces (MM API) and tool environments. This is a problem for the publishing industry because the investment for a multimedia title is very high.

The driving force behind the CD technology was and will continue to be the consumer market. End-user prices of below $400 for a CD-ROM player and starting with $800 for a CD-I system with high quality DA sound and optional full motion video support can only be reached if the quantities are comparable with today's CD-DA players, i.e. much higher than PC quantities. The key factor for a customer to buy such equipment besides the price is the availability of several interesting applications with real added value. Publishers are willing to invest in new technologies if the business is large enough (consumer market) and the infrastructure (standards, authoring tools, knowledgeable developers) is in place. Large business perspective justifies the investment in sophisticated authoring tools, which is the precondition for involving creative title developers. Both the manufacturer of consumer products and the publishing industry has an essential interest in developing sophisticated development tools for CD titles.

Several end-users are interested in professional and consumer multimedia titles, especially for training and education. They are willing to train at home but they would like to have equivalent help support if they are doing the business. E.g. recently an insurance company decided to develop a training application based on CD-I, to pay the price difference between a CD-I and a CD-DA player for the employees if they would use the system at home and they decided to integrate comparable on-line help into future releases of their business applications.

This paper describes the integration of CD-I systems into professional UNIX and X-Windows/Motif based workstations. It is based on results of a feasibility study we performed in the framework of the ESPRIT project MULTIWORKS (MULTimedia Integrated WORKStation, project no. 2105 and 2713). The Multiworks high-end workstation is a 486 PC with an EISA bus, a Super VGA display system, ISDN WAN

and Ethernet (also FDDI) LAN connections. A basic requirement for the integration of multimedia CD-technology in Multiworks was to support off-the-shelf CD-I titles running in a Motif window concurrent with other UNIX applications and to support new UNIX/Motif multimedia applications, which use the CD-I player as a digital video, sound and CD-ROM sub-system.

2. CD-I characteristics

CD-I is a family of hardware and software components [3] that covers not only the delivery of audio and visual information, but also a CD volume and file format standard. The heart of the CD-I system is the Multimedia Controller (see Fig. 1), based on an MC68070 microprocessor on which the real-time operating system CD-RTOS is running. The current family of stand-alone CD-I devices (CD18x series) can easily be equipped with various communication paths (RS232, SCSI, Ethernet) to interface the system to peripherals or other computers. All time critical operations in CD-I are supported by special hardware (e.g. the video system that is configurable for PAL or NTSC or the audio processing and output controller), thus enabling many visual effects, moving image display, advanced graphics functions and audio delivery in different quality levels. Interactivity is supported by homogeneously integrated input devices like mice, keyboard and optional extensions like graphics tablet, joy-stick and others. CD-I offers four video planes that can be mixed in various ways: a cursor plane, a foreground and a background digital video plane and a background plane that can either show a constant colored background or an external video signal. Video information (i.e. still images or image sequences) can be encoded in five different schemes:

- differential encoding for luminance (Y) and two chrominance components (U, V) for high image quality, called DYUV.
- Color Lookup Table encoding (CLUT8, CLUT7, CLUT4) for computer graphics.
- run-length encoding for animation, which can compress CLUT coded images further.
- natural representation in red, green and blue color components with five bits per pixel (RGB555).
- (optional) full motion video decompression based on a subset of the MPEG ISO standard [4, 5].

CD-I offers four different audio quality levels: CD-DA (which in stereo mode uses 100% of the bandwidth of about 150 kbits/sec.), ADPCM level A (HiFi quality stereo, data rate 50%), level B (FM quality stereo, data rate 25%) and level C (AM quality stereo, data rate 12,5%). Depending on the encoding level up to 16 mono or 8 stereo audio channels can exist simultaneously in the same CD-I track.

The CD-RTOS operating system is an extended version of OS-9 specifically tailored to CD-I. Because of its modular design the basic system is stored as a ROM module, additional software modules (e.g. I/O handler) can be loaded at runtime. The inherent real-time capabilities guarantee proper handling of all time critical events and opera-

tions, while the operating system kernel and system services provide a programming interface very closely resembling to UNIX.

A standard CD-I application is a program (e.g. written in C) developed for the CD-RTOS operating system that can be played on any stand-alone CD-I system that fulfills the CD-I base case requirements [6]. Such a program is normally stored on a CD-I disk.

The multimedia capabilities of the CD-I hardware are used and controlled by using the native CD-I application function calls from the CD-I Presentation Support Library (Native CD-I API = NAPI) [3, P. 230ff]. These functions are an integral part of the CD-I system; the functions, their parameters and the data structures they interact with are described in the GREEN BOOK [6]. Since these functions are elementary, a high level Application Programming Interface (CD-I API) and development environment, called BALBOA [7], is available on top of the NAPI. BALBOA is organized into a group of manager (library functions), each handles a CD-I element, such as video display, animation, sound, cursor controls, timer or combined multimedia objects.

3. Requirements on the Integration of CD-I in UNIX / X-Windows

As described above, it should be possible to start and control standard CD-I titles fully executed on the CD-I subsystem by using the keyboard and mouse of the UNIX workstation exclusively. But it should also be possible to develop new multimedia UNIX applications that use the CD-I subsystem as a high performance video, sound and CD-ROM player that offers functions like:

- special visual effects, e.g. continuous dissolves, wipes, curtain and mosaic effects.
- output of audio signals in different quality levels, ranging from CD digital audio quality to mono AM broadcast quality.
- playing short audio segments (soundmaps) from memory.
- instantaneous switching between audio channels during a running application.
- full true color representation (e.g. effectively 3 x 8 bits = 16 million colors simultaneously when using the DYUV image coding method, or 256 out of 16 million colors for the CLUT8 coding method).
- arbitrarily shaped image regions for matte functions.
- instantaneous switching between two video planes.
- full screen full motion video decompression.

This requires on the host side, beside the communication aspect,

- the support of digital video overlay functions (controller) and their integration in the window system,
- mouse, keyboard and screen redirection. If the mouse cursor is moved to the CD-I

window, the coordinates have to be mapped to the full CD-I screen size and transmitted with possible keyboard input to the CD-I subsystem.

- a high level multimedia application programming interface (MM API) based on a high level multimedia peripheral interface (HPI). All real-time critical functions must be executed on the CD-I subsystem by using the HPI functions. The host HPI handlers have to manage the execution of these functions or a sequence of these functions.
- CD-ROM file service support. It should be possible to access database or text objects stored on a CD, mounted on the CD-I system, by standard UNIX volume and file operations. This requires the mapping of ISO 9660 structures, read from the CD-I system as physical blocks, to UNIX objects on the host system.
- down loading of CD-I system software.

To respond to multimedia events on the CD-I subsystem quickly enough and to read large data objects in reasonable time from a CD, a high speed physical connection between the host and the CD-I system is required, which is normally SCSI.

4. Architecture

The architecture of the distributed approach (Fig. 2) is based on the client / server paradigm, with the host being the client and the CD-I subsystem being the server. Fig. 3 shows the components and their interrelationships relevant for multimedia host applications. A major design goal was to support in a later step an integrated CD-I solution based on a (PC) plug-in controller with all CD-I specific hardware / software components and connected to a standard CD-ROM drive. Therefore it was necessary to layer the system in such a way that this integrated approach can as much as possible be based on components of the distributed approach without change of at least the upper software layer. This results in the usage of standard computer communication interfaces and protocols. Three groups of new software components can be identified in Fig. 3:

- modules on the client and server side implementing the communication service and providing the HPI at the client side. These are the "Remote Procedure Call" layer (RPC), Transport layer (TL), Data Link layer driver (streams modules, file manager and SCSI drivers) and the Server Process (SP) on the CD-I side. Most of these components have to be changed or can be dropped in the integrated approach.
- components implementing the High Peripheral Interface (HPI) at the server side. This consists of the CD-I API Library, which is normally the BALBOA library with extensions, and a support module (CD-I API Extensions) that permits the installation of time critical application specific routines (callbacks) on the CD-I side at runtime (dynamic linking, etc.).
- the Multimedia Application Interface (MM API) Layer on client side. This set of high level library functions and widgets available in the programmer's development kit is realized by (a sequence of) HPI function calls.

The envisaged method of interaction between host and CD-I subsystem is as follows:

a. The application program on the host invokes a multimedia function (HPI function) like "create image" or "play real time record" by calling the appropriate entry of the RPC client library at the client side. This includes the delivery of several parameters like image coding method, file name on the CD etc.

b. The RPC component converts this request into a message which is then sent to the CD-I system by the communication software.

c. The communication parts on the CD-I side receive the message and hand it over to the RPC server component, which runs under the control of the server process.

d. The RPC routines decode the message and do the appropriate call to the HPI, i.e. to the CD-I API or to the CD-I API Extensions, respectively.

e. The HPI functions are mapped to the low level operations of the native CD-I API and many time critical actions are done in the context of the CD-I API components and CD-RTOS (by virtue of the BALBOA software). Application-specific time critical routines are loaded on top of the HPI at the CD-I side (application specific callbacks).

f. Return values are sent back to the application stepping backward through the same components through which the call was propagated.

Since the host waits for the completion of the operation it has requested from the CD-I device, this interaction mode is called a "synchronous service request". In case of "asynchronous service requests" an acknowledgement value is returned to the application, but after that, the function continues to operate at the CD-I side. Examples are "play real time record" or "start animation sequence". There may be a need to send a signal back to the application, when the started operation terminates or on similar events. This can be done by the server calling a callback routine that the client has specified before.

For understanding of the characteristics of communication between the UNIX host and the CD-I subsystem, it is useful to list the different cases where information has to be passed between both systems:

- exchange of control information and small amounts of data (remote procedure call).
- parallel virtual input channels for the CD-I device mouse and keyboard.
- exchange of massive amount of data (especially read operations from the CD disk).
- audio stream from CD-I device.
- video stream from CD-I device.

These logical communication channels have to be mapped onto physical communication media with regard to the expected transferred data volume and the tolerable transmission delays. Apart from the implementation effort, it would be desirable to have a single digital communication path (e.g. a parallel bidirectional data bus) that carries all the types of data. However, since the video and audio streams available at the output connectors of the current (table top) CD-I systems are both analog, they have to be kept separate from the rest. Therefore an analog video channel is connected

to the analog video input connector of the digital video overlay (DVO) board and a twin analog channel for stereo audio is connected with a separate stereo amplifier or headset. In a future integrated approach the video (and audio) interface to the host system must be digital.

The purpose of the Transport Library is to provide a software interface for the transparent sending and receiving of messages (byte blocks) while hiding the details of the used physical medium (SCSI, Ethernet) as well as the related protocol layers below. Such lower protocol layers are responsible for synchronization, error detection and handling, packaging etc. Since the connection between host and server is a point-to-point link, the software needed will be simple (e.g. no network layer). However, multiple virtual channels must be supported at least for the transmission of remote procedure calls in parallel with keyboard input and mouse events regarding the CD-I window.

The term "Remote Procedure Call" denotes a concept that allows one process (the client process) to have another process (the server process) execute a procedure call. Client process and server process are separate and may even reside on different machines. From the viewpoint of the client process, the remote procedure call behaves exactly like a local procedure call, i.e. as if the procedure was executed in the client process's own address space. To implement a remote procedure call, several types of data have to be exchanged between both processes (i.e. have to be transferred across the underlying communications medium or network):

- control information (identification of client and server processes, identification of the procedure to be called etc.),
- the input arguments passed to the called procedure and
- the result of the procedure, passed back to the client process.

Since the representation of data types (byte order, word length) may be different on different machines, the arguments and results to be transferred have to be transformed into a universal representation. Basis for the RPC implementation was the public domain RPC package developed by Sun Microsystems [8]. This package uses a general scheme for encoding RPC arguments, the XDR external data representation scheme, which is also able to transfer complex data structures including e.g. data referenced by pointers. The RPC package consists of two main components:

- an RPC stub routine generator (rpcgen) that processes a formal C-like specification of the data types and routines to be implemented by RPC. The rpcgen pre-compiler generates the source code of the client and server stub routines and associated header files also XDR routines for the transformation of those data types defined in the specification file.
- a set of library functions that are called by the stub routines on client and server side (for simplification not drawn in Fig. 3).

The client stub routines library resides between the HPI and the RPC library at the client side. It contains a stub routine for each HPI function and an XDR routine for

each user-defined parameter type, while the proper RPC library is the generic RPC/XDR part. Similarly, the RPC library at the server side forms the generic part, while the RPC/XDR stub routines reside between the server process and the HPI at the CD-I side.

The intention of defining an HPI is to simplify applications programming and to remove real-time interactions from the interface between client and server. This requires a set of generic high level functions handling often used multimedia effects like cut, fade in / fade out, dissolve (cross fade) wipes and blends. Most of these effects are fully supported by primitives of the native API of CD-I. A prerequisite for these functions is to describe the properties and relations of audio-visual "objects" in a way that permits the server program to perform autonomously much detail work that otherwise would have to be done by the application programmer. The abstraction from machine details, a major goal of the HPI, means that some very specialized effects that are possible for a standalone CD-I device cannot be realized in the host-server configuration. An overview of HPI functions and a detailed discussion about their realization in a client/server environment is given in [7 and 9]. Not only the intention to provide an abstract machine, but also some limitations in the real-time behavior, caused by the communication necessary between both machines prevent the programmer of a client/server application from doing everything that is possible on a standalone CD-I system.

An important aspect of the definition of the HPI is the way real-time events are handled. The standard way of achieving fast communication between different application parts and processes on a CD-I device is the usage of signals. This mechanism is used for instance for informing the application process that the play-back of an audio track (which is played asynchronously, i.e. only triggered by the application) has finished. Other examples are timer events, generation of signals on video blanking intervals and others. In the client/server environment the only way of handling time-critical events is to do the processing of signals entirely on the CD-I server side, i.e. the client has to instruct the server to link specific action routines to the occurrence of signals. This means the server has to be configurable and extensible at application runtime by user-provided routines. This may happen in two ways:

a. the host downloads a module of type "Program" that can be executed (chained or forked) by the server as a subprocess. This module can also be loaded directly from the CD-I disk.

b. the host downloads a subroutine module that is dynamically linked to the server and is executed in the context of the server process.

The server process on CD-I side represents in principal the host application on the server side. Under the assumption that the host runs a multiuser/multitasking operating system, the user can have more than one application started simultaneously. He can switch between them at his will. Similarly, it may be desirable to start several multimedia applications. Of course, only the one which currently has the input focus, will be active (running) producing audio/video output. If concurrent multimedia applications should have access to the same CD-I player, a listener - server model has

to be implemented on the CD-I system. The listener process listens for incoming connection requests from host applications. When a request arrives, the listener spawns a server process that services the application. The listener continues awaiting further connection requests. Such a configuration requires that the complete hardware and software context be automatically changed when the user switches between applications, i.e. when the corresponding server process is changed. Whether the CD-I software below the native CDI API (Green Book) does support this, requires further investigations.

In the model described so far, the host system triggers the CD-I system to perform specific operations with the multimedia material stored on the CD-I disk. However, many possible applications (e.g. travel information systems, sales support systems) also need direct access to the data stored on the CD-I disk (airline time tables, order numbers, prices and many more). Therefore it is necessary to have access to the files on the disk from the host. This requires the integration of the CD file access into the file system structure of the host in a homogeneous way. For UNIX this means that the file and directory structure of the CD should become a part of the normal UNIX file system tree.

It has already been mentioned that the current implementation covers the integration of the CD-I video output into the X windowing environment using an analog video path. The analog CVBS signal is fed into digital video overlay board, also a Multiworks result. The display parameters like image size (upscaling and downscaling is possible), position on the screen, brightness, contrast and color saturation can be controlled by the X application using the call interface that has been provided by the VEX (video extensions for X) working group. If, additionally, live video input from a camera, television receiver, VCR or videodisc is needed, these signals can be fed into the external video input jack of the CD-I system. The external video signal is then displayed in the CD-I backdrop plane, i.e. behind the CD-I generated screen contents (the video planes A and B of the CD-I system can be set transparent for the CD-I screen also for arbitrarily shaped regions).

Fig. 4 shows the global architecture and the integration of the above described components in the UNIX WS.

The architecture described so far reflects the situation that the multimedia application is running on the UNIX workstation and the CD-I subsystem is used as multimedia peripheral. A major requirement is also to support standard CD-I titles executed on the CD-I system but controlled by the host input devices and displayed in a window of the host monitor. If the host mouse is moved or a mouse key is pressed, a corresponding mouse event is buffered in an event queue of the X window system. The output of the CD-I system will be shown in a window on the host screen. As long as the cursor location is inside this window, mouse events are transmitted to the CD-I system, where they cause a CD-I signal to be generated or simulated. The mouse coordinates transmitted are relative to the CD-I window's origin and in units of the CD-I coordinate system. The continuous transfer of mouse events to the CD-I system

is needed for two classes of operations:

- the mouse movements control real-time video or graphics operations (drawing programs, or video games where visual objects are to be moved across the screen).
- some instantaneous actions have to be performed depending on the current mouse position ('hot spots').

Considering the handling of (host) keyboard input, the situation is very similar. When the CD-I window has the input focus, the keyboard events are redirected to the CD-I system. Fig. 5 shows the global architecture of such a configuration.

5. Conclusions

The above described concepts were evaluated in a prototype implementation. The results are positive especially regarding performance and the usability. The efficiency of the HPI functionality was tested by the development of a UNIX demonstration application. It became obvious that multimedia applications which are based only on the functions of the host HPI and so do not use any application specific call back routines on the CD-I subsystem, are easy to develop. Moreover it was proved that the architecture supports general multimedia applications as well those in which time critical procedures are executed on the CD-I system. But in this case the realization also showed that the development process of multimedia applications grows much more complex, as the programmer has to think and work in two very different development environments. So the question arises whether the commercial benefit of playing consumer titles in professional workstations and so offering binary compatibility to the GREEN BOOK can compensate for these disadvantages. If this requirement is eliminated and the CD-I hardware concept (chip set) is realized e.g. on a PC board and if the CD-I API is fully integrated into the host operating system or window system, the authoring process will be essentially simplified. The integration of the CD-ROM XA in UNIX / Motif is based on this concept and realized in the product XGRIP 3.0 which has been proven successful. The shortly specified concept of the "Bridge Disc" by Philips and Sony takes this problem into account in so far as the same CD can be used on a CD-I and a CD-ROM XA system if the specific application programs are developed for both platforms.

6. Acknowledgements

Significant ideas realized in the prototype of CD-I integration in UNIX / Motif were developed by A. Wepner and Dr. R. Mesters, both authors of the feasibility study report. Particularly the experiences A. Wepner gained during the implementation of major parts of the system software and the test application were essential improvements of the concept. This applies also for many constructive contributions of B. Klee and W. Kramer during the design phase. The author would like to record his thanks for these contributions, for the commitment during the implementation and for the constructive comments on this paper.

7. References

[1] Inteco Europe Corp., Surrey GU21 1JD, UK;
Multimedia in Europe;
Volume 1, Section F; July 1991

[2] Meißner, K.;
Aspects of Multimedia Application Support in UNIX;
Multimedia in Action, Fifth International Protext Conference;
Luxembourg; November, 20 -22, 1991

[3] Philips International;
Compact Disc-Interactive: A Designer's Overview;
Kluwer Technical Books, Deventer-Antwerpen; 1987

[4] Dorerty, R.;
Full-Motion Video Technique for Interactive CD Player;
Electronic World News; June 17, 1991; P. 12

[5] Sijstermans, F.; Meer, J. van der;
CD-I Full-Motion Video Encoding on Parallel Computer;
Communication of the ACM; Vol. 34, No. 4; April 1991; P. 81 - 91

[6] N. V. Philips; Sony Corp.;
Compact Disc Interactive Media Provisional Specification (GREEN BOOK);
May 1986

[7] Philips International;
BALBOA Run-Time Environment: Programmer's Guide
August 1990

[8] Sun Microsystems;
Network Programming, Revision A;
May 9, 1988

[9] Mester, R.; Wepner, A.;
Feasibility Study: Integration of CDI Devices into a Multiworks Workstation;
ESPRIT Project 2105, Multiworks; Document Id. PKI9006BP8.3;
September 30, 1990

8. Trademarks:

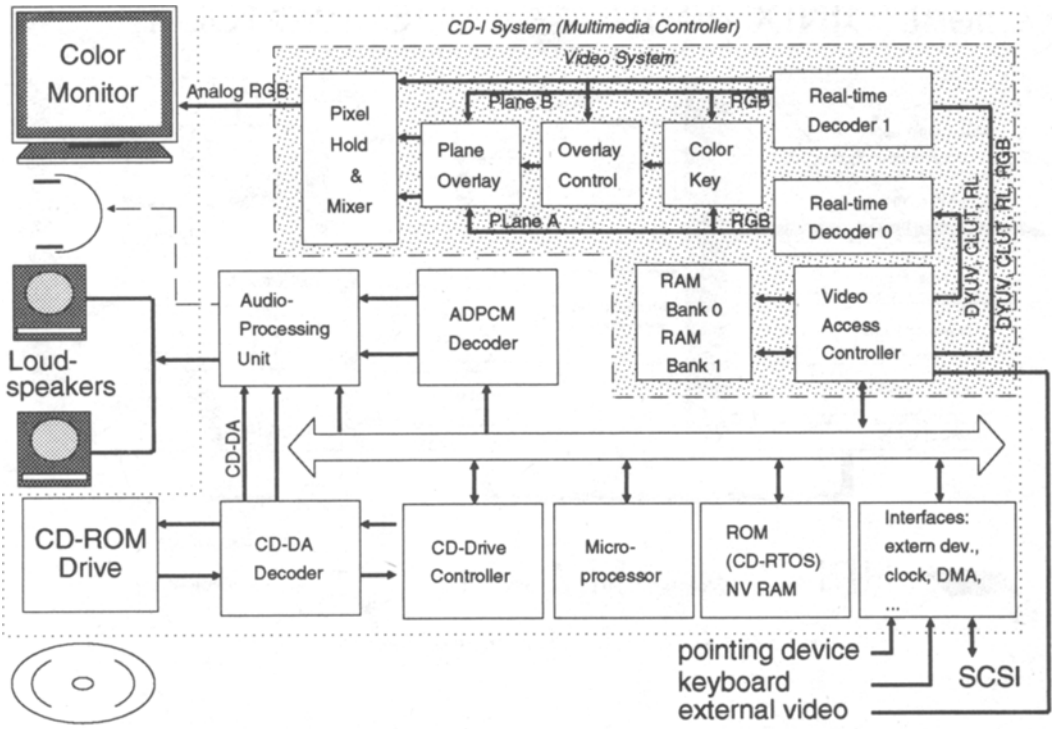

Fig. 1: Hardware components of a CD-I system [3]

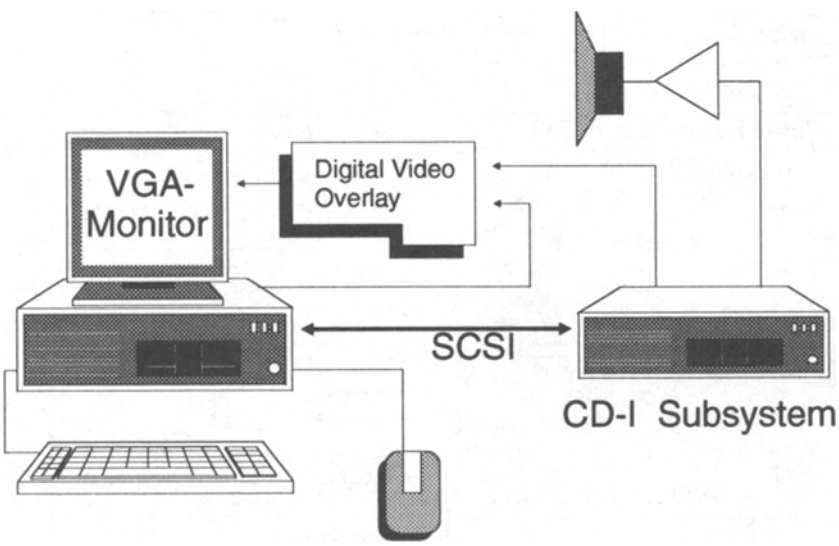

Fig. 2: System Configuration of the Prototype

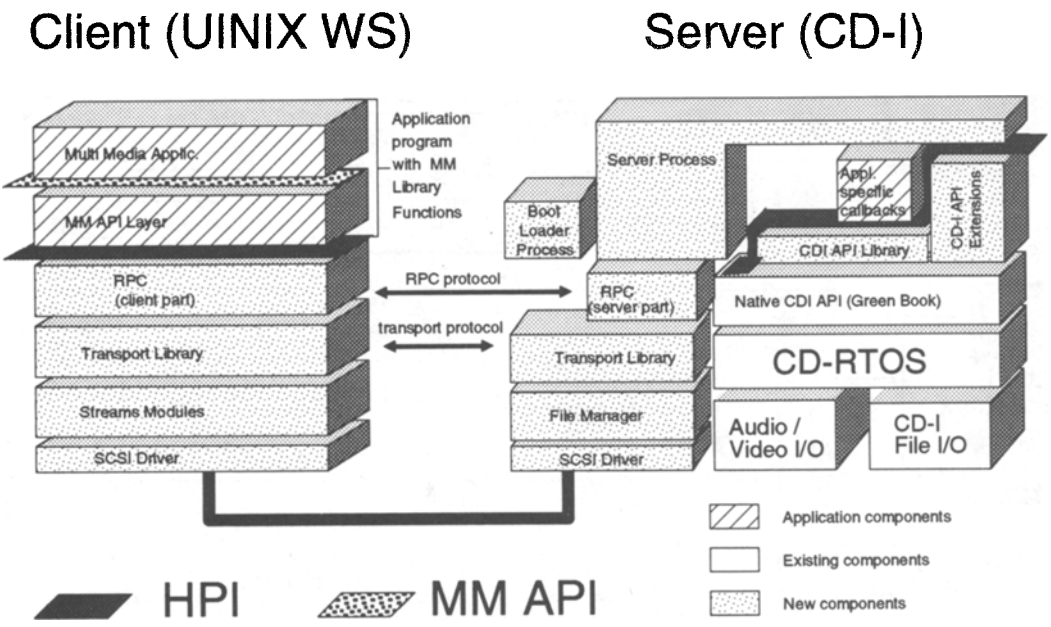

Fig. 3: Client / Server Architecture [9]

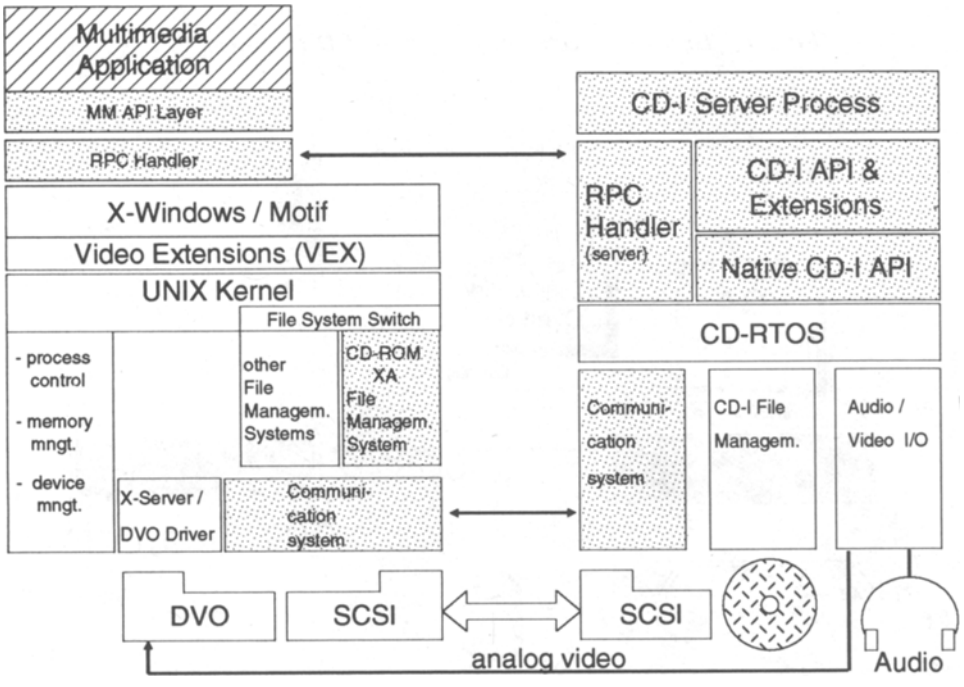

Fig. 4: Architecture with MM Host Applications

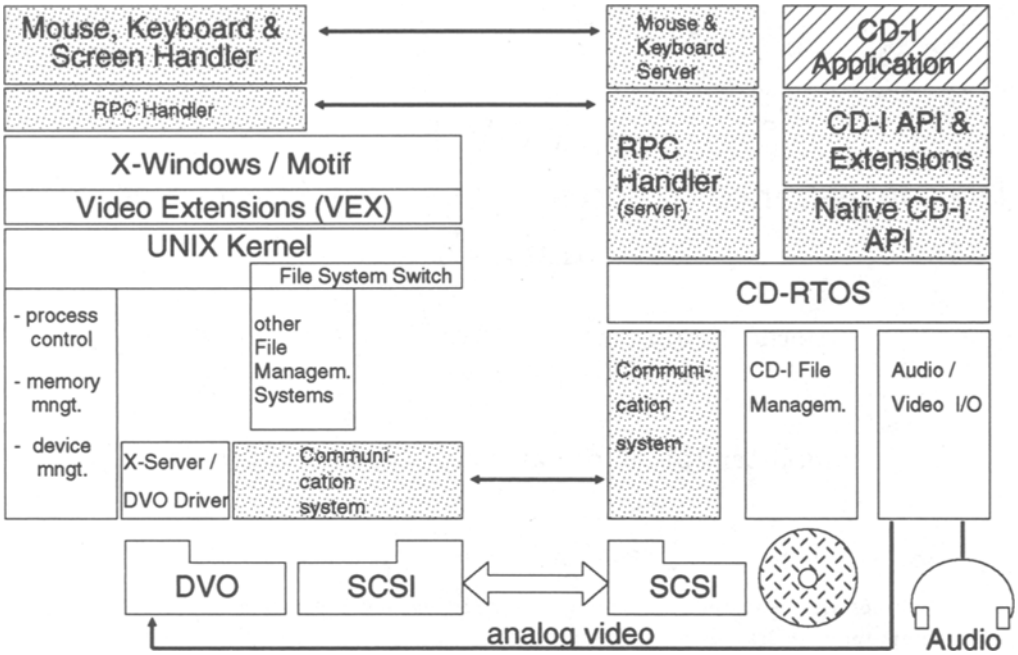

Fig. 5: Architecture with MM CD-I Applications